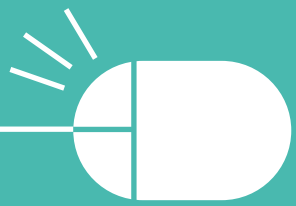
So einfach!

Eigene Spiele erstellen mit Roblox

TEIL 1

Erste Schritte mit Roblox Studio

1 Roblox Studio



1.1 Roblox-Account

1.2 Studio installieren

1.3 Vorlagen

1.4 Steuerung

1.5 Teile/Parts

1.6 Workspace

1.7 Testen und spielen

1.8 Spiele speichern oder veröffentlichen

1.9 Effekte

Wenn du schon Roblox gespielt hast, weißt du, dass die Welt von Roblox schier grenzenlos ist. Es gibt auf dieser **Online-Spieleplattform** nicht nur Millionen von Spielern, sondern auch Millionen von Spielen! Neben professionellen Studios können auch die aktiven Spieler selbst Spiele erstellen, die wiederum von anderen Benutzern gespielt werden können. Manche verdienen sogar richtig viel Geld dabei! Denn es gibt auch ein *internes Währungssystem*, **Robux**, das für Spielerpässe oder Items genutzt werden kann. Und mit dem **Roblox Studio** gibt es in Roblox eine eigene *Spieleengine*, mit der jeder eigene Spiele gestalten kann. Und sowohl Roblox, das Spiel, als auch das Roblox Studio sind zudem noch kostenlos!

Kinder und Jugendliche als Entwickler

Wusstest du, dass die meisten Spiele, die sich auf Roblox befinden, von Kindern und Jugendlichen hergestellt wurden? Natürlich sind diese oft nicht so bekannt und beliebt wie *AdoptMe!*, *Jailbreak* oder *Work at a Pizza Place*, aber trotzdem haben es diese Kinder geschafft, eigene Ideen umzusetzen!

Eigene Spiele erstellen

Auch du kannst eigene Roblox-Spiele erschaffen! Die *Basics* dazu lernst du hier in diesem Buch an mehreren Beispielprojekten.

Auf Seite Seite 86/87 werden die Spiele, die du mithilfe des Buchs programmieren kannst, kurz vorgestellt!

WICHTIG

Während du Roblox auf PC, Handy, Konsole oder Tablet spielen kannst, funktioniert das Roblox Studio *nur* für Windows oder Mac! In diesem Buch werden alle Beispiele in Windows gezeigt, du kannst aber auch mit einem Mac arbeiten, lediglich die Steuerung und Tastenkombinationen sind ein bisschen anders!

ROBLOX

Der Name »Roblox« setzt sich aus den englischen Wörtern *robot* (Roboter) und *blocks* (Blöcke) zusammen.

1.1 Roblox-Account

Für das Roblox Studio brauchst du, genau wie für Roblox auch, einen **Roblox-Account**. Wenn du schon einen Account zum Spielen hast, kannst du diesen auch für das Roblox Studio nutzen.

Account erstellen

Einen Account kannst du im Internet auf der Webseite von Roblox erstellen: *www.roblox.com*

Hier kannst du einen selbst gewählten **Benutzernamen** eingeben, ein neues **Passwort** dafür sowie dein **Geburtsdatum**. Am besten erstellst du den Account gemeinsam mit deinen Eltern!

1 Benutzername

Hier gibst du natürlich *nicht* deinen richtigen Namen an, auch keinen Vornamen oder irgendeinen Hinweis, der Rückschlüsse auf deine echte Person geben könnte!

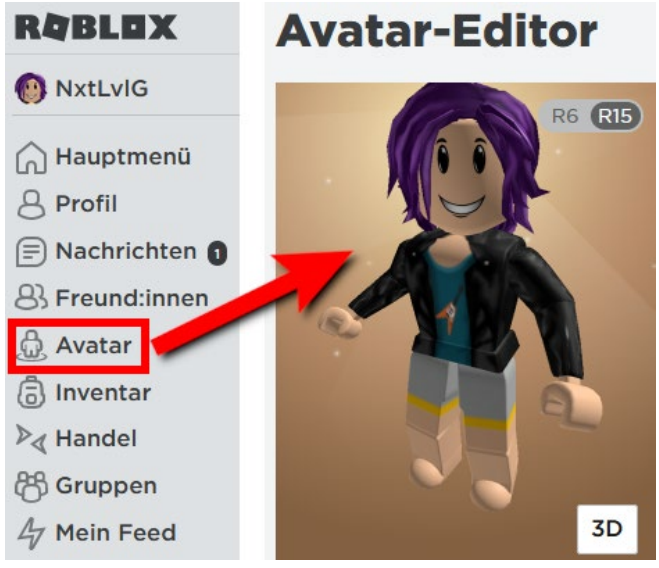
Überlege dir einfach einen coolen, lustigen oder verrückten Namen! Beliebte Namen wie *Superman* oder *Arielle* sind natürlich schon vergeben, aber du kannst ja auch Nummern oder Sonderzeichen einbauen. Überlege dir einfach etwas Kreatives!

TIPP

Du kannst deinen Namen in den Einstellungen deines Accounts auch später noch mal verändern!

2 Der Avatar-Editor

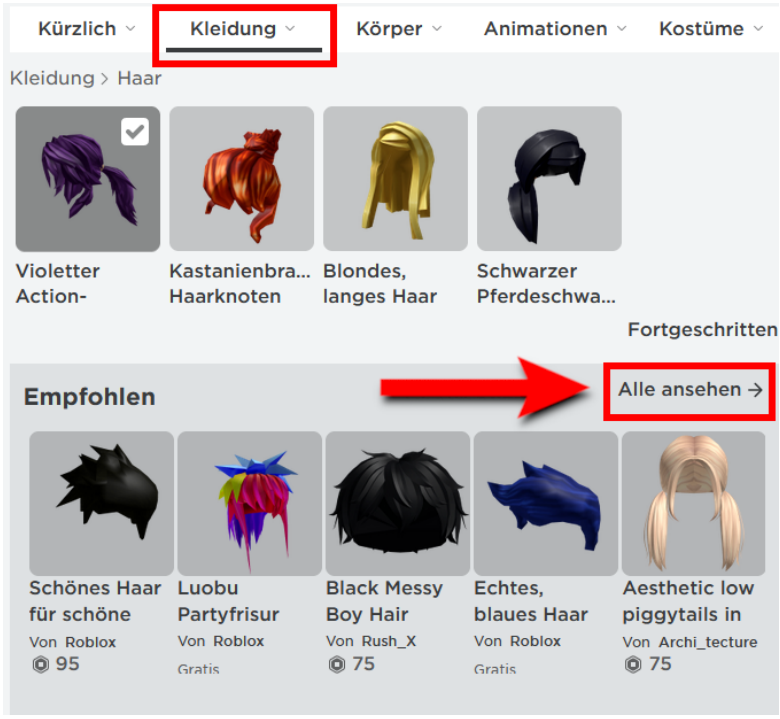
Jetzt solltest du noch deinen Avatar anpassen, denn der voreingestellte ist ziemlich langweilig! Klicke einfach links am Rand auf **Avatar**, und der **Avatar-Editor** öffnet sich.



3 Avatar anpassen

Im Avatar-Editor kannst du alles anpassen, die Kleidung, die Haare, die Figur und die Hautfarbe. Natürlich kannst du auch coole Accessoires ergänzen, wie Sonnenbrillen, Hüte oder lustige Tiere, die auf der Schulter sitzen können! Probiere einfach aus, was dir gefällt!

Viele Objekte kosten **Robux**, die *interne Währung*, aber es gibt auch eine Menge kostenloser Sachen!



TIPP

Zurück kommst du am einfachsten, wenn Du wieder auf den Avatar-Button klickst!

Wenn dein Avatar fertig ist und du mit Roblox als Spiel ein bisschen vertraut bist, können wir loslegen!

1.2 Studio installieren

Öffne zunächst Roblox, das Spiel, entweder im Internet auf *www.roblox.com* oder aus deinen **Apps** bzw. **Programmen** und logge dich mit deinem Account ein. Dann klickst du ganz oben in der Menüleiste auf **Erstellen**, und das Roblox Studio wird installiert.

Es dauert ein paar Minuten, und das Roblox Studio öffnet sich, jetzt kannst du mit der **Spieleprogrammierung** beginnen!

In diesem Kapitel lernst du die Basics im Roblox Studio. Also wie man **Modelle erstellt** und bearbeitet, **coole Materialien** verwendet und das **Spiel testet** und richtig speichert. Am Ende von diesem Kapitel kannst du deine neu erworbenen Fähigkeiten gleich ausprobieren und ein paar coole Modelle nachbauen, zum Beispiel eine **Schatztruhe** oder einen **Zaubertrank**!

Deutsch und Englisch

Roblox Studio ist seit 2024 auch auf Deutsch erhältlich. Trotzdem sind Untermenüs und Befehle weiterhin auf Englisch. Aber auch, wenn du noch nicht viel Englisch verstehst, lernst du die für dich wichtigsten Begriffe sicher schnell. Und einige davon kommen dir vielleicht schon aus anderen Computerspielen bekannt vor.

WICHTIG

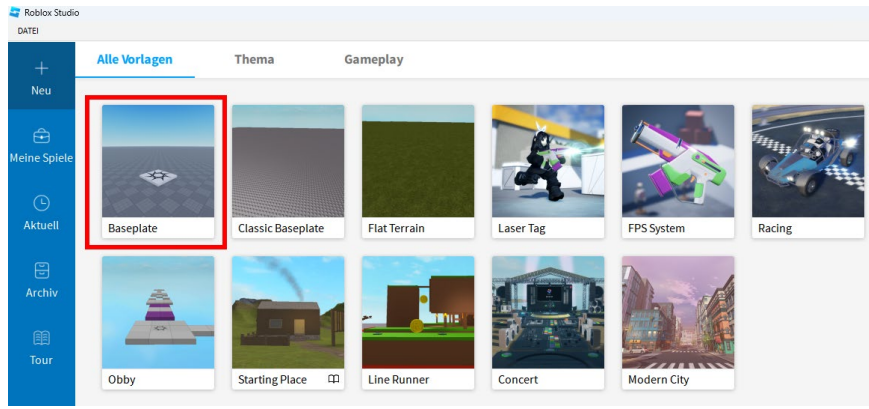
Du brauchst eine Internetverbindung, wenn du das Roblox Studio nutzen möchtest!

1.3 Vorlagen

Vorlagen kannst du gut als Einstieg für eigenen Spiele benutzen. Die meisten Vorlagen beinhalten schon vorgefertigte Elemente, um Spiele zu erstellen.

Vorlage auswählen

Öffne das Roblox Studio, es befindet sich in deinen Apps oder Programmen. Nun kannst du eine **Vorlage** auswählen. Wähle die **Baseplate Vorlage** aus.



Der Game Editor

Jetzt öffnet sich ein neues Fenster. In der Mitte befindet sich das **Game-Editor-Fenster**. Hier baust du später dein Spiel, bearbeitest deine Map und kannst eigene 3D-Modelle bauen.

Die Ausgabe

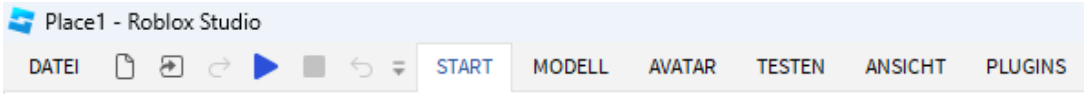
Unter dem Game-Editor-Fenster befindet sich die **Ausgabe**. Hier werden Statusberichte und Fehlermeldungen von Scripten angezeigt. Mehr dazu erfährst du in Kapitel 4 »Programmieren mit Lua«.

SCRIPT

Ein Script enthält kurze Anweisungen und Befehle, was der Computer machen soll. Ähnlich wie bei einem Kochrezept ist die richtige Reihenfolge der Anweisungen wichtig. In Roblox wird natürlich nichts gekocht, aber du kannst mit einem Script zum Beispiel die Farbe von Steinen ändern oder auch die Geschwindigkeit deiner Spielfigur.

Die Menüleiste

Oben siehst du die Menüleiste mit den verschiedenen **Registerkarten**.



Dateien öffnen und speichern

In der Menüleiste kannst du unter **Datei** Dateien öffnen oder speichern.

Vor und Zurück

Daneben findest du auch die beiden Pfeile für Vor und Zurück. Mit den Pfeilen kannst du deine letzten Aktionen also rückgängig machen oder wiederherstellen.

Alternativ kannst du für Zurück auch die Tastenkombination Strg + Z drücken und für Vor die Tasten Strg + Y.

Weitere Funktionen

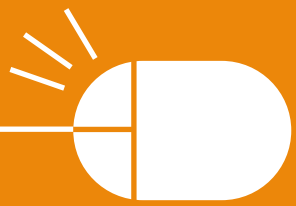
Wenn du auf die Registerkarten, also **Start**, **Modell**, **Testen**, **Ansicht** oder **Plugins** klickst, öffnen sich neue Untermenüs mit weiteren Funktionen. Bei der Spieleprogrammierung arbeiten wir hauptsächlich in der Registerkarte **Start** oder **Modell**.

FENSTER SCHLIESSEN

Du kannst alle Andockfenster wie *Eigenschaften*, *Ausgabe* oder den *Explorer* über das kleine X schließen. Wenn du die Fenster wieder benötigst, kannst du sie in der Registerkarte **Ansicht** wieder aktivieren. Aber Achtung! Wenn du das Game-Editor-Fenster schließt, wird die aktuelle Datei komplett geschlossen!

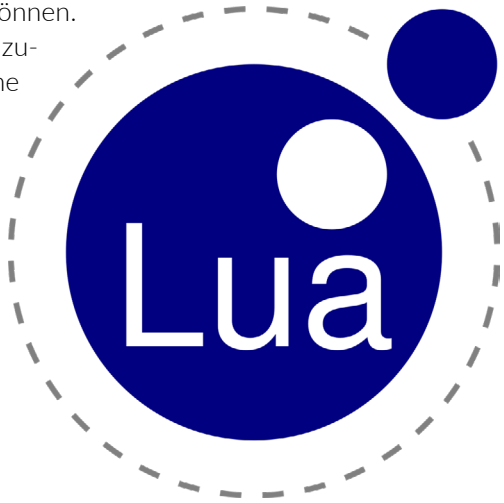
Start	Wichtigste Funktionen aus **Modell**, Test und Allgemein
Modell	Alles zum Bauen und Bearbeiten
Testen	Testen und spielen
Ansicht	Ein-/Ausblenden der Andockfenster (Explorer etc.)
Plugins	Zusätzliche Erweiterungen z.B. für Animationen

4 Programmieren mit Lua



In diesem Kapitel schreiben wir kleine Scripte, die im Spiel verwendet werden können. Die Scripte machen das Spiel *interaktiver*. Wir können damit Teilen **Funktionen** zuweisen, sie beispielsweise umfärben oder verschwinden lassen. Auch wenn deine Spielfigur Werkzeuge oder Waffen benutzt, wird das über Scripte gesteuert.

Die Scripte sind also eine Art Liste mit Befehlen und Anweisungen, was der Computer machen soll. Ähnlich wie ein Kochrezept oder eine Lego-Bauanleitung. Die richtige Reihenfolge der Befehle ist wichtig, und jeder kleine Schritt muss genau beschrieben sein!



Was ist Lua?

Roblox verwendet eine etwas modifizierte Form der **Scriptsprache Lua**. Der Name bedeutet auf Portugiesisch *Mond*, denn Lua wurde in einem Land entwickelt, wo portugiesisch gesprochen wird, nämlich in Brasilien.

Viele **Computerspiele** besitzen Scripte, die in **Lua** geschrieben sind, weil sie wenig Speicherplatz verbrauchen und blitzschnell sind. Und Lua-Scripte lassen sich einfach in andere Programme einbinden. Genau das nutzen wir bei Roblox Studio.

WICHTIG

Ohne Programmieren geht es leider nicht! Lies dir dieses Kapitel aufmerksam durch. Du musst nicht alles gleich auf Anhieb verstehen, aber du bekommst einen guten Eindruck, wie Programmieren funktioniert und kannst dann die Scripte, die wir später für die Spiele verwenden, besser verstehen!

4.1 Das erste Script

1 Teil erstellen und umbenennen

Erstelle ein neues Teil (Part) und benenne es um in *Testblock*. Es ist besser, du gibst den Teilen, die du verbaust, eindeutige Namen, dann findest du sie schneller im Workspace wieder. Alle **Teile**, die erstellt werden, heißen dort nämlich immer nur *Part*, und wenn du ein bestimmes Teil verändern willst, wird es schwierig, es zu finden.

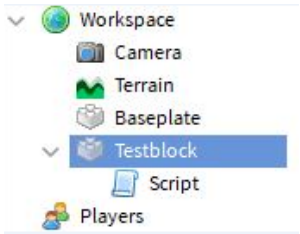
Um den Block umzubenennen, klickst du ihn mit der rechten Maustaste an und wählst **Umbenennen**. Du kannst den Namen auch in den **Eigenschaften** unter **Name** verändern.

UMLAUTE

Verwende am besten *keine* Umlaute (ä, ö, ü), weder bei den Teilen noch in Scripten, denn Umlaute gibt es in der englischen Sprache nicht, und sie werden auch nicht in Programmiersprachen verwendet! Sie können sogar **Fehler** im Programm verursachen.

2 Script erstellen

Klicke nun auf das kleine Plus neben *Textblock* im Workspace und erstelle ein **Script**. Wenn du alles richtig gemacht hast, sieht es im Workspace nun so aus:

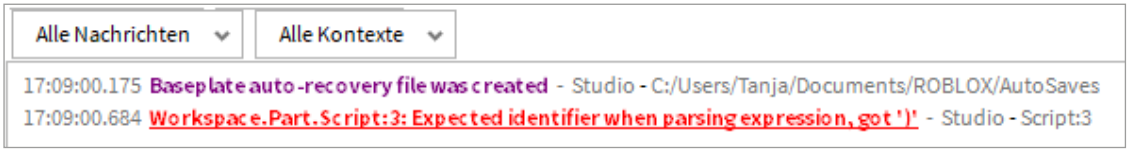


3 Script starten

Wenn du das Script anklickst, öffnet es sich. Es besteht nur aus einer Zeile, nämlich:

```
print("Hello world!")
```

Schon diese eine Zeile ist ein komplettes Script! Wenn du in der Menüleiste auf **Testen|Spielen** klickst und das Spiel startest, siehst du in der **Ausgabe** die Nachricht »Hello world!« Eventuell musst du dafür im **Ausgabe**-Fenster nach oben scrollen! Wenn das **Ausgabe**-Fenster nicht offen ist, aktiviere es in der Registerkarte **Ansicht**!



4 Text ändern

`print` bedeutet auf Englisch »drucken«, und genau das macht das Script. Es druckt den Text in das **Ausgabe**-Fenster! Gedruckt wird das, was zwischen den Anführungszeichen steht.

WICHTIG

Wichtig ist, dass sich das Script eingeschoben im Testblock befindet, denn nur dann gehört es wirklich zum Testblock und kann diesem Anweisungen geben. Man nennt das eine *Hierarchie*, das Script ist dem Testblock untergeordnet.

In der Programmierung bezeichnet man den Testblock als *Parent* (Elternteil) und das untergeordnete Script als *Child* (Kind). So wie in einer echten Familie können die Eltern natürlich auch mehrere Kinder haben!

CHALLENGE

Ändere den Text im Script in einen eigenen, lustigen Text!

4.2 Die erste Funktion

Der Text unseres ersten Scripts wird momentan immer direkt zu Beginn des Spiels ausgegeben.

Wenn du nun möchtest, dass der Text nur dann ausgegeben wird, wenn etwas, also zum Beispiel der Spieler, den *Testblock* berührt, kannst du eine **Funktion** dafür schreiben.

```
function SagWas() 1
    print("Wer berührt mich da?") 2 3
end 4

script.Parent.Touched:Connect(SagWas) 5 6
```

1. `SagWas` ist der Name der Funktion. Der Name ist frei wählbar und hat keinen Einfluss auf das, was der Computer tun soll. Du könntest die Funktion also auch *Banane* nennen, für den Computer sind es zunächst nur einfach Buchstaben!
2. `print("Wer berührt mich da?")` schreibt den Text in den Klammern in die Output-Konsole.
3. Der Befehl `print` ist etwas *eingerückt* und steht nicht direkt bündig unter der Funktion. Das erleichtert die Lesbarkeit des Programmcodes, so weiß man gleich, dass der Befehl zur Funktion gehört! Alles, was eingerückt zwischen `function` und `end` steht, gehört also zur Funktion.
4. `end` beendet die Funktion.
5. `script.Parent` bedeutet *Elternteil von diesem Script*, und das ist der Testblock!
6. `Touched` ist das **Event** (Ereignis), das heißt, wenn irgendetwas den Block berührt (Touch = berühren). Das Event `Touched` wird über den Doppelpunkt und die Roblox-eigene Funktion `Connect` (verbinden) mit der Funktion `SagWas` verbunden.

FUNKTION

Eine Funktion ist ein kleiner, in sich geschlossener Abschnitt im Programmcode, den du von jeder beliebigen Stelle in deinem Programm aus aufrufen kannst. Du kannst bestimmen, wann und wie oft die Funktion ausgeführt werden soll. So wird dein Programmcode viel übersichtlicher!

WICHTIG

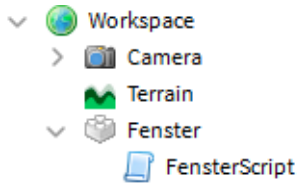
Übrigens, wenn der Testblock direkt auf der Bodenplatte liegt oder nicht verankert war (**Anker**) und diese zu Spielbeginn berührt, löst die Bodenplatte ebenfalls die Funktion aus, und der Text wird im **Ausgabe**-Fenster ausgegeben!

4.3 Die Basics

Jetzt wird es etwas trocken! Es ist kein Problem, wenn du nicht alles sofort verstehst, du lernst ja auch viel, während du die Spiele entwickelst, denn in jedem Spiel in diesem Buch wird programmiert!

Einfache Pfade

Wenn du in einem Script auf ein bestimmtes Teil/Part zugreifen möchtest, musst du dem Computer mitteilen, welches Part du meinst und wo sich das Teil befindet. Du musst also den **Pfad** vom Teil angeben.



Durchsichtige Fenster

Hier befindet sich ein Teil/Part namens *Fenster* im Workspace. Und in diesem Part befindet sich ein Script, das *FensterScript*. Wenn man das Teil *Fenster* etwas durchsichtig machen möchte, kann man im Script diesen Pfad verwenden.

1 Teil erstellen

Baue ein Teil namens Fenster und füge dort ein Script ein. Im Script löscht du den Vorgabetext und schreibst stattdessen diese *eine Zeile* Programmcode:

```
game.Workspace.Fenster.Transparency = 0.9
```

Wichtig sind die **Punkte** zwischen den Wörtern! Und der **Pfad** beginnt immer mit `game`. Deine Teile schreibst du exakt so, wie du sie benannt hast.

WICHTIG

Denk daran, den Teilen/Parts, die du benutzt, immer **eindeutige Namen** zu geben! Wenn alle Objekte im Workspace nur »Part« heißen, weiß der Computer natürlich nicht, welches Teil nun gemeint ist! Und auch du findest dich so in der langen Liste vom Workspace besser zurecht.

TEIL 2

Eigene Spiele erstellen

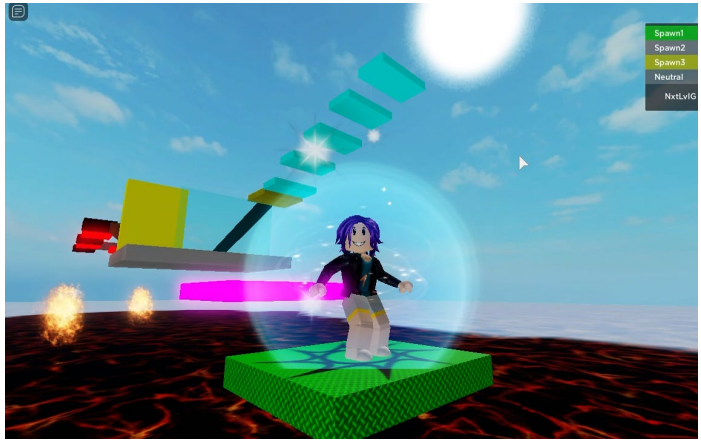
Nachdem du das Roblox Studio kennengelernt hast, kannst du anfangen, die in den folgenden Kapiteln beschriebenen *vier spannenden Spiele* nachzubauen.

Bei allen Spielen steht der **Schwierigkeitsgrad** dabei sowie der Aufwand für die Scripte und die Modellierung. Die wichtigsten **Features** sind ebenfalls aufgelistet. Plane dir genug Zeit ein, um ein Spiel zu entwickeln, das kann ein paar Stunden dauern!

WICHTIG

Bei den Spielen werden leichte Grundkenntnisse in Roblox Studio vorausgesetzt, diese besitzt du in jedem Fall, wenn du den ersten Teil dieses Buchs aufmerksam gelesen hast!

Obby

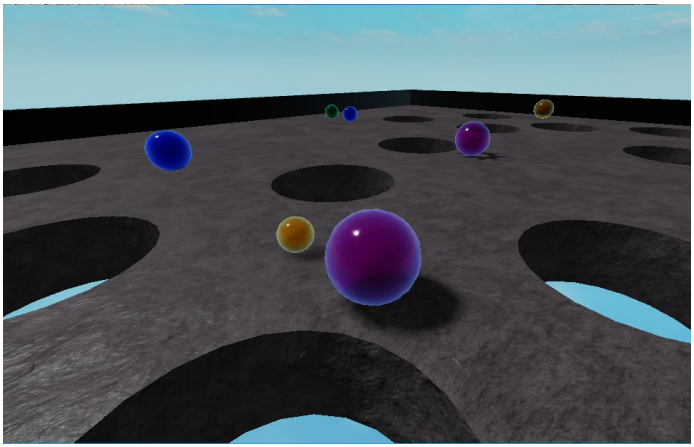


SCHWIERIGKEIT	1
DAUER	1 – 2 Std.
SCHRIPTE	
MODELLIERUNG	
FEATURES	Viele interaktive Blöcke und kleine Scripte, die auch gut in anderen Spielen eingesetzt werden können!

Escape Game

SCHWIERIGKEIT	1
DAUER	1 Std.
SCHRIPTE	
MODELLIERUNG	
FEATURES	Wettlauf gegen die Zeit, First Person Camera, Ziel spawnt an zufälliger Stelle, Soundeffekte!

Murmelspiel



SCHWIERIGKEIT	2
DAUER	1,5 Std.
SCRIPTE	
MODELLIERUNG	
FEATURES	Murmel als Spielfigur, nur ein kleines Script, aber mehrere Attachments!

Autorennen

SCHWIERIGKEIT	3
DAUER	ab 2 Std.
SCRIPTE	
MODELLIERUNG	
FEATURES	Coole Autos aus der Toolbox, Start-Ampel mit echtem Licht, Zeitmessung und SpeedUps mit Effekten!

Coole Features

Hinten im Buch im Kapitel 9 »Coole Features« findest du zusätzliche Ideen, wie **Punktezähler**, **Collectibles** oder **computergesteuerte Spielfiguren**, die du in jedes deiner Spiele einfügen kannst.

Du kannst natürlich die Ideen und Scripte der einzelnen Spiele zu neuen Spielideen zusammenfügen! Die *Start-Ampel* vom **Autorennen** kannst du auch im **Obby** ergänzen oder ein paar der *funktionalen Blöcke* aus dem **Obby** beim **Escape Game** einbauen, und die *Soundeffekte* aus dem **Escape Game** passen in jedes Spiel!